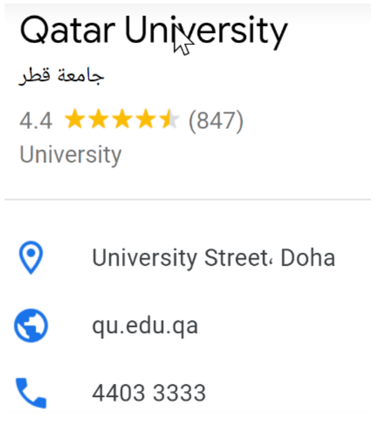
Google Maps

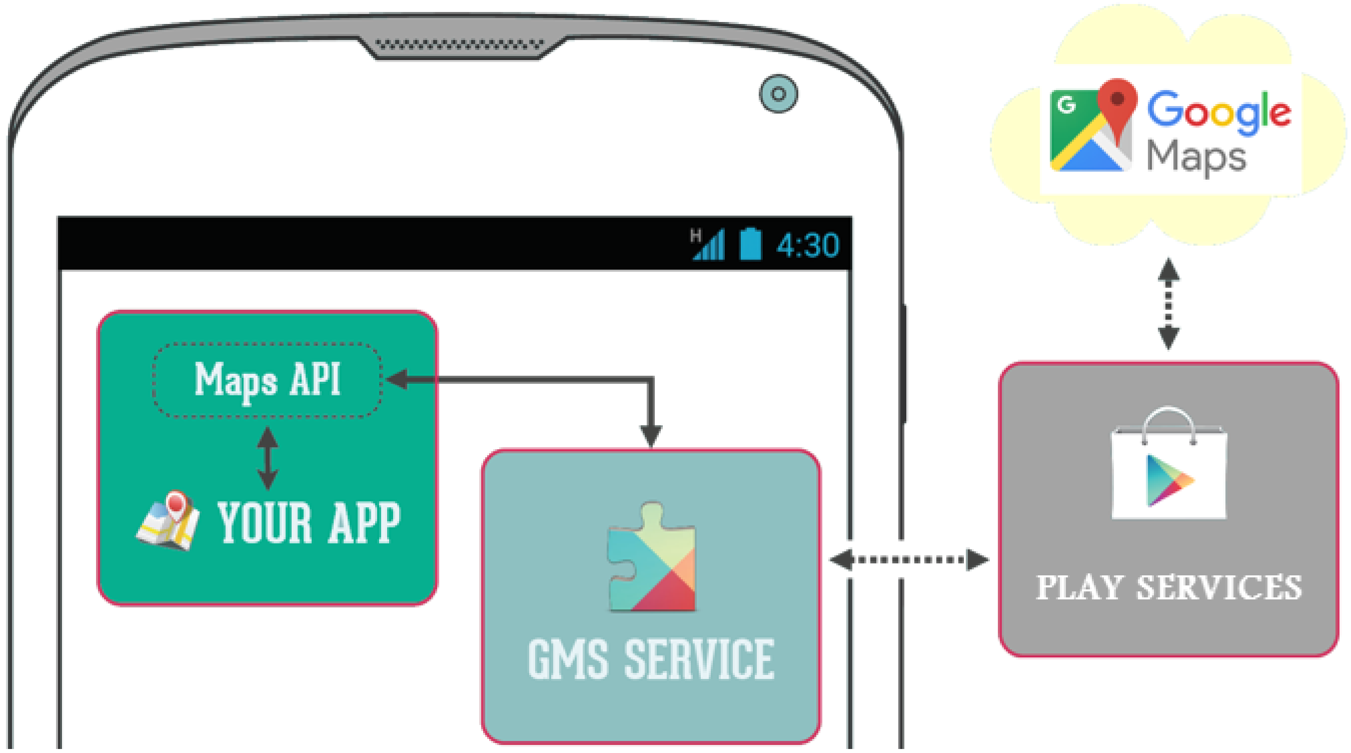
Google Maps Platform Key Services

- **Maps**:
  - Apps can integrate customized and interactive maps, satellite imagery and Street View imagery
- **Routes**:
  - Allow users to find the best <u>route</u> to get from A to Z using public transport, biking, driving, or walking.
  - Compute travel times and distances
  - Real-time traffic updates about the selected route
- **Places**:
  - Users can search details about million **points of interest** around the world including place names, addresses, images, contact information and reviews

College of Engineering - Qatar University
كلية الهندسة - جامعة قطر
Lat: 25.37499, Long: 51.49066
Qatar University
Admission
Tennis Co
Pizza
College of Engineering - Qatar University
كلية الهندسة - جامعة قطر


Qatar University
جامعة قطر
4.4 ★★★★★ (847)
University
University Street، Doha
qu.edu.qa
4403 3333

Google Mobile Services (GMS)



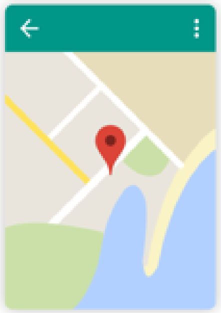
- Add these dependences to build.gradle

```
// Map & Location Services
implementation 'com.google.android.gms:play-services-maps:18.0.0'
implementation 'com.google.android.gms:play-services-location:18.0.0'
implementation 'com.google.maps.android:maps-ktx:2.3.0'
implementation 'org.jetbrains.kotlinx:kotlinx-coroutines-play-services:1.5.2'
```

Typical Programming Tasks in Location-aware App

- Visualise data in a **custom map**
- Get the device **geolocation** (latitude & longitude)
- **Geocoding**: finding the GPS coordinates of an address
  - E.g., what are coordinates of Qatar University
- **Reverse Geocoding**: finding the address of a GPS coordinates
  - E.g., what is the address at
- **Location tracking** as the user moves
  - Uber track current location during the ride
- **Geofencing**: trigger an action/notification when the device is in area of interest
  - E.g., switch on the coordinator light when the user approaches the area of their home

Display a Map using MapView



- Use MapView component to display and interact with Google Maps

- Need to add Google Maps API key to *res/values/strings.xml* file
  - More details on how to get the API Key https://developers.google.com/maps/documentation/android-sdk/get-api-key

```
<string name="google_maps_key">AIzaSyC6f0s...</string>
```

Customize Map

College of Engineering - Qatar University
كلية الهندسة - جامعة قطر
Lat: 25.37499, Long: 51.49066
Qatar University
Admission Registration
College of Engineering - Qatar University
Tennis Co
Pizza

- The displayed map can be **customized** such as:
  - Add marker
  - Add overlay (e.g., image over the map)
  - Change the zoom level
  - Handle events such as Point of Interest (PoI) click event

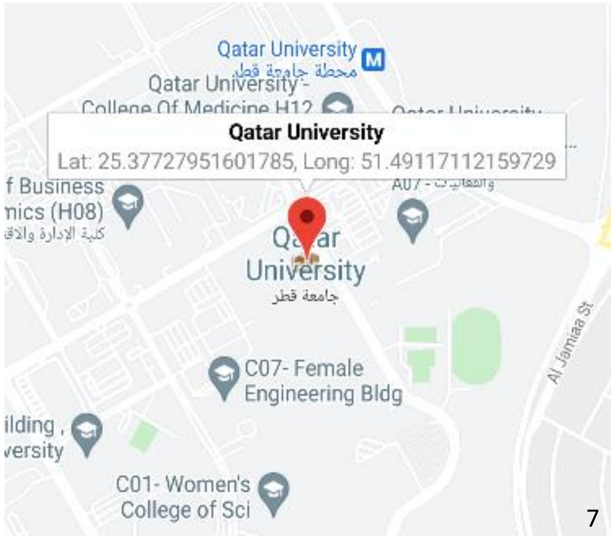
```
// MapScreen
LaunchedEffect(mapView) {
    val googleMap = googleMap.awaitMap()
    mapViewModel.googleMap = googleMap
    mapViewModel.onMapReady(location)
}
```

```
/** MapViewModel
 * Manipulates the map once available.
 * This function is called when the map is ready to be used.
 * This is where we can add markers or lines, add listeners
 * or zoom to a location.
 */
fun onMapReady(location: Location) {
    zoomToLocation(location)
    addMarker(location)
    addOverlayImage(location)
    setOnPoiClick()
    setOnMapLongClick()
}
```

Add Marker

- Marker identify a location on the map at a particular geo coordinates
  - When the marker is clicked an **info window** displays the marker's title and snippet text

```
fun addMarker(location: Location) {
    // A Snippet is a text displayed below the title
    val snippetText = "Lat: ${location.latitude}, Long: ${location.longitude}"
    val latLng = LatLng(location.latitude, location.longitude)
    googleMap.addMarker {
        position(latLng)
        title(location.name)
        snippet(snippetText)
    }?.showInfoWindow()
}
```

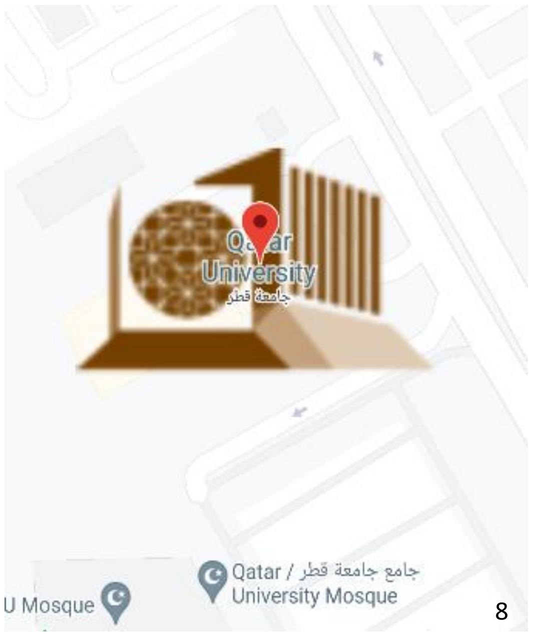

Qatar University

Lat: 25.37727951601785, Long: 51.49117112159729

Add Overlay

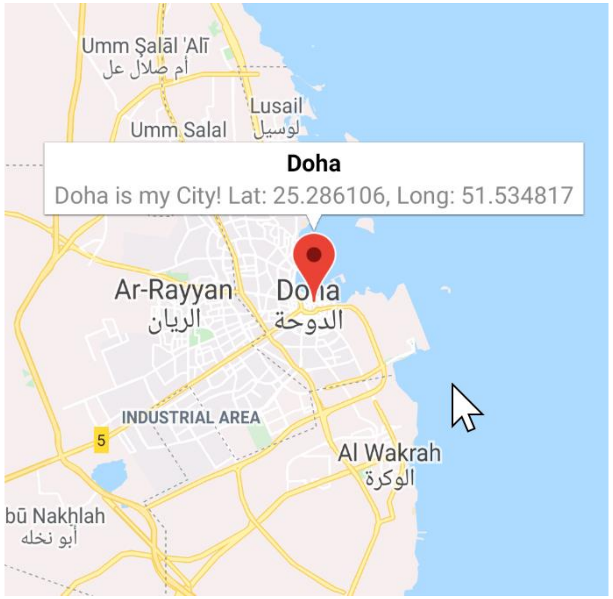
- A ground **overlay** is an image that is displayed over the map at a particular geo coordinates
  - Overlays **size** and **orientation** changes when rotating, tilting or zooming the map

```
fun addOverlayImage(location: Location, overlaySize: Float = 100f, resourceId: Int = R.drawable.qu_logo) {
    val latLng = LatLng(location.latitude, location.longitude)
    // Add overlay image at the specified location
     googleMap.addGroundOverlay {
         position(latLng, overlaySize)
         image(BitmapDescriptorFactory.fromResource(resourceId))
    }
}
```



Zoom to a Location

- Zoom to a Location by moving the map view to a particular **geo coordinates** and change the zoom level
- Zoom level values:
  - 1: World
  - 5: Continent
  - 10: City
  - 15: Streets
  - 20: Buildings



Doha
Doha is my City! Lat: 25.286106, Long: 51.534817

Zoom level 10

```
fun zoomToLocation(location: Location, zoomLevel: Float = 15f) {
    val latLng = LatLng(location.latitude, location.longitude)
    val cameraUpdate = CameraUpdateFactory
                                    .newLatLngZoom(latLng, zoomLevel)

    googleMap.moveCamera(cameraUpdate)
}
```

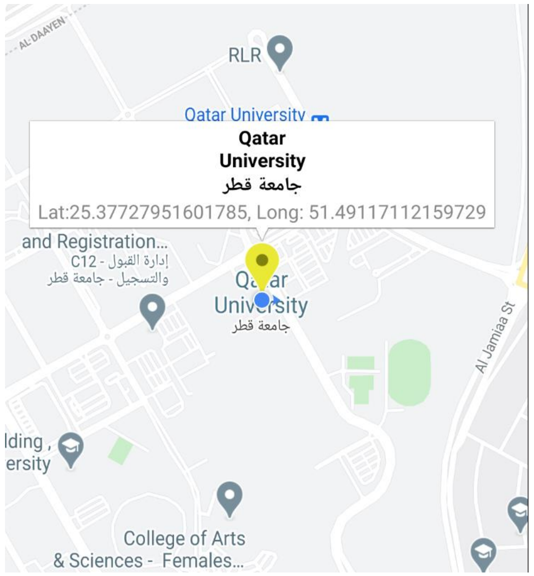
Other Map Customization

- Set the map type

```
googleMap.mapType = GoogleMap.MAP_TYPE_NORMAL       // OR

googleMap.mapType = GoogleMap.MAP_TYPE_HYBRID       // OR

googleMap.mapType = GoogleMap.MAP_TYPE_SATELLITE    // OR

googleMap.mapType = GoogleMap.MAP_TYPE_TERRAIN
```

Handle Point of Interest (PoI) click event

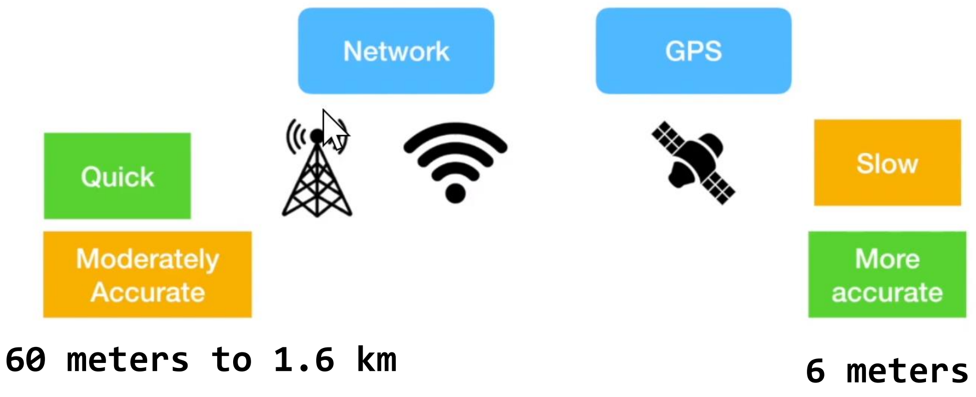
- If you want to respond to a user tapping on a PoI, you can use `googleMap.setOnPoiClickListener`
    - poi parameter has the **placeId**, **name** and **geo coordinates** (i.e., `latitude` & `longitude`)



```
googleMap.setOnPoiClickListener { poi ->
    println(">> Debug. Clicked PoI placeId: ${poi.placeId}. Name: ${poi.name}")
    // A Snippet is Additional text that's displayed below the title.
    val snippet = "Lat:${poi.latLng.latitude}, Long: ${poi.latLng.longitude}"

    poiMarker = map.addMarker {
        position(poi.latLng)
        title(poi.name)
        snippet(snippet)
        icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_YELLOW))
    }
    poiMarker.showInfoWindow()
}
```

Get User Location

- Request last known location of the user's device
  - Location is determined by the **LocationServices** using WiFi & Cellular Tower and/or GPS (Global Positioning System)



```
val fusedLocationClient =
    LocationServices.getFusedLocationProviderClient(appContext)
val lastLocation = fusedLocationClient.lastLocation.await()
lastLocation?.let {
        val currentLocation = "Lat: ${it.latitude} & Long: ${it.longitude}"
        println(">> Debug: $currentLocation")
    }
}
```

Request location updates

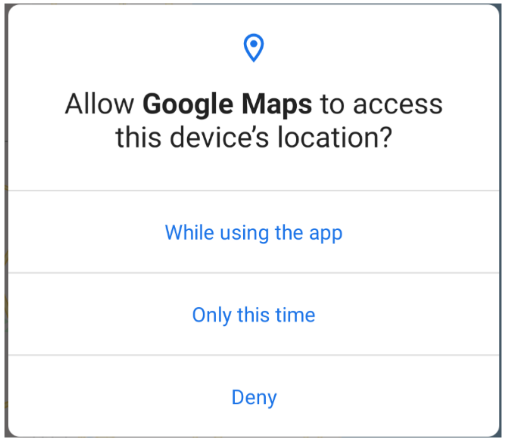
- To get the location (latitude and longitude) of the device at regular intervals you can use `fusedLocationClient.requestLocationUpdates`
  - The location provider invokes the `LocationCallback.onLocationResult()` on a regular interval. The incoming argument contains a list `Location` object containing the location's latitude and longitude

```
fun startLocationUpdates() {
    val locationRequest: LocationRequest = LocationRequest.create().apply {
        interval = 10000 // every 10 seconds
        priority = LocationRequest.PRIORITY_HIGH_ACCURACY
    }
    fusedLocationClient.requestLocationUpdates(
        locationRequest, locationCallback
    )
}
private val locationCallback = object : LocationCallback() {
    override fun onLocationResult(locationResult: LocationResult?) {
        locationResult ?: return
        locationResult.locations.forEach {
            deviceLocation = LatLng(it.latitude, it.longitude)
            println(">> Debug: Lat: ${it.latitude} & Long: ${it.longitude}")
        }
    }
}
```

Request Location Permission

- At runtime must ask for the permission to access the device's location using
  ```
  rememberLauncherForActivityResult(
      ActivityResultContracts.RequestPermission())
  ```

```
// Register request permission callback, which handles the user's response to the system permission dialog
private val requestPermissionLauncher = rememberLauncherForActivityResult(
          ActivityResultContracts.RequestPermission())
    // Callback for the result from requesting permission
    { isGranted: Boolean ->
        if (isGranted) {
            // Permission is granted. Enable My Location button on the map
            mapViewModel.enableMyLocation()
        }
    }
...
// Ask for the permission to access the user’s device location
// The registered call back gets the result of this request
requestPermissionLauncher.launch(Manifest.permission.ACCESS_FINE_LOCATION)
```


Allow **Google Maps** to access this device’s location?

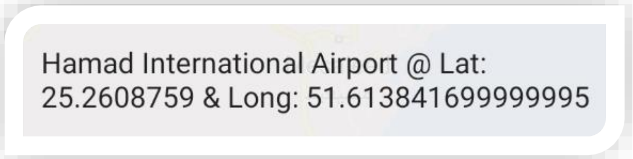
While using the app

Only this time

Deny

Geocoding

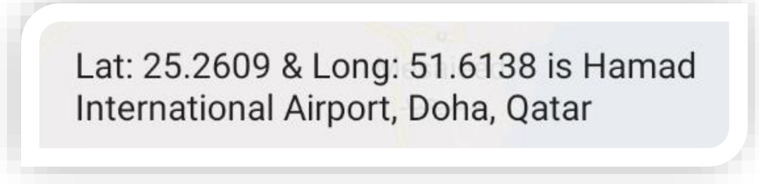
- **Geocoding** is the process of converting an address (e.g., location name or a street address) into geographic coordinates (lat, lng), which you can use to place markers on a map, or zoom to that location on the map



Hamad International Airport @ Lat: 25.2608759 & Long: 51.613841699999995

```
/*
    Geocoding = converting an address or location name (like a street address) into
    geographic coordinates (lat, lng)
*/
 private fun getGeoCoordinates(locationAddress: String): GeoLocation? {
    val geocoder = Geocoder(this)
    val coordinates = geocoder.getFromLocationName(locationAddress, 1)
    return if (coordinates != null && coordinates.size > 0) {
         val latitude = coordinates[0].latitude
         val longitude = coordinates[0].longitude
         GeoLocation(latitude, longitude)
    } else {
         null
    }
 }
```

Reverse Geocoding



Lat: 25.2609 & Long: 51.6138 is Hamad International Airport, Doha, Qatar

- **Reverse geocoding** is the process of converting geographic coordinates (lat, lng) into a human-readable location address

```
/*
    Reverse geocoding = converting geographic coordinates (lat, lng)
    into a human-readable location address
*/
fun getLocation(lat: Double, lng: Double): Location? {
    val geocoder = Geocoder(appContext)
    val locations = geocoder.getFromLocation(lat, lng, 1)

    return if (locations!= null && locations.size > 0) {
        val name = locations[0]?.featureName ?: ""
        val city = locations[0]?.locality ?: ""
        val country = locations[0]?.countryName ?: ""
        Location(name, city, country, lat, lng)
    } else {
        null
    }
}
```

Resources

- Android Google Maps Codelab
    - https://codelabs.developers.google.com/codelabs/advanced-android-kotlin-training-maps
- Google Maps Android samples
    - https://github.com/googlemaps/android-samples
- Receive location updates in Android with Kotlin Codelab
    - https://codelabs.developers.google.com/codelabs/while-in-use-location/
- Adding geofencing to your map Codelab
    - https://developer.android.com/codelabs/advanced-android-kotlin-training-geofencing